Design Brief: Group 5

Matthew Mifsud, Luca Vella, Graham Pellegrini, Julian Falzon

March 10, 2024, Document v.7839

This documentation outlines the design of a project utilizing the LPC4088 microcontroller to interpret telephone keypad presses, also known as Dual-Tone Multi-Frequency (DTMF) signals. The LPC4088 will sample and process these signals, enabling the display of results on an LCD and feedback through two status LEDs. Additionally, users will be able to configure the display state using a switch.

1 Introduction

Dual-tone multi-frequency (DTMF) signaling is a method used in telecommunication systems to transmit digits or symbols over telephone lines. It utilizes pairs of audio frequencies to represent each symbol, which include the digits from 0 to 9, the letters A to D, and special characters such as * and #. Each symbol is encoded using a unique pair of frequencies, consisting of one low and one high frequency. Importantly, these frequency pairs are chosen from within the voice frequency range, making them audible. [1].

Freq	1209Hz	1336Hz	1477Hz	1633Hz
697Hz	1	2	3	A
770Hz	4	5	6	B
852Hz	7	8	9	C
941Hz	*	0	#	D

Table 1: Frequency Mappings

The project assigned aims to build a DTMF decoder utilising an ARM Cortex-M4 microcontroller, capable of processing line-level audio inputs with different signal characteristics. This includes handling inputs with perfect timing (fixed duration), inputs with imperfect timing (small random variations in duration), and noisy, imperfect inputs. The system will then decode these signals and display the results on an external LCD screen in real-time.

The design is structured to proceed through a defined sequence of steps. The following documentation aims to present a comprehensive overview of the design process by briefly going through each step in the sequence to ensure a thorough understanding of the approach to the problem.

2 System Design

2.1 Hardware Components

Line output to ADC Circuit (input): A DTMF audio source will be sent to the device from a computer by using the headphone jack line output as input to the ADC. The audio signals used for testing and implementation will be those sourced from VLE as .mp3 files with variation in signal properties. However, since line output signals vary nominally between approximately -1V to 1V and the wanted ADC signal to be processed is needed to vary from 0V to VREF (where VREF is the working voltage from the microcontroller LPC4088) then an intermediate circuit needs to be built [2].

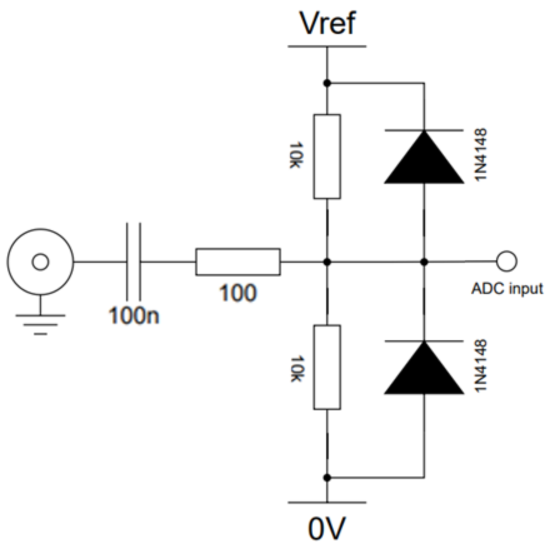

Figure 1: Line output to ADC

Breaking the circuit down:

1. Firstly, the audio line output signal is passed serially through a 100nF capacitor which acts as an open circuit for the DC component of

the signal. Thereby, allowing only the AC component of the signal to pass.

2. Secondly, a serial 100-ohm resistor is used to simply limit circuit current.
3. At the first junction a voltage divider containing two 10K-ohm resistors are used [3]. Thus, halving the input voltage range spanning from VREF to ground and placing the signal within the suitable range for the ADC.
4. A parallel second junction to the voltage divider of two respective 1N4148 diodes is used to protect the circuit from overvoltage (falling outside the ADC range) [4]. If one input exceeds the ADC range then the opposite polar diode will begin to conduct and negate the excess voltage.
5. Finally an ADC appropriate signal is achieved.

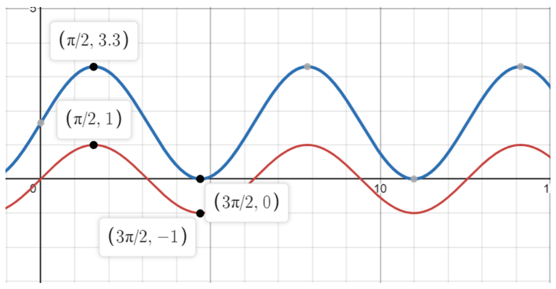

Figure 2: Line and ADC signals

Therefore, the circuit described above serves as an input conditioning stage for the audio signal before it enters the WM8731 codec. The ADC in the WM8731 converts the analogue audio signal into digital format, which can then be processed digitally by a microcontroller. The values produced by the on board codec can be directly read through the memory-mapped I/O on the LPC4088 microcontroller board.

User Interaction Switch: A switch will be used to facilitate user interaction with the system. This includes modifying system settings to enable/disable scrolling and clearing the LCD. This will be done through the GPIO pins which connect the switch to the LPC4088 microcontroller, which in turn monitors the state of the switch and takes appropriate actions based on user input.

Liquid-crystal display: The Hitachi HD44780-based 16x2 LCD is interfaced with the LPC4088 microcontroller using GPIO pins for data signals, control signals, and power connections (VCC and GND) [5]. The microcontroller will send commands and data to the LCD in order to display the decoded output to the user.

Feedback LEDs: Two LEDs connected through the GPIO pins, will provide visual user feedback on the system state. A green LED will indicate that the system is in a safe defined state, whilst a red LED will indicate an error.

Note: All additional hardware components where sourced by Computer Engineer through the electronics supplier 'Fabian Enterprises Ltd'

2.2 Interrupt-Driven Sampling

The system will be multithreaded through eulating two threads, a read and a decode (main) thread, through the use of a timer that which calls an interrupt handler that triggers the ADC to sample the input signal at a frequency of at least 3266Hz. The sample rate was chosen based on the Nyquist theorem, which states that the rate should be at least twice the highest frequency of the signal, for accurate measurement.

2.3 Buffer store

Each captured sample is immediately stored in a circular buffer on the system's RAM which overwrites the oldest data once full capacity has been reached.

2.4 Functional Requirements

Requirement: Solution must be real-time; reacting to one or more inputs that can change at any time.

Description: The system will constantly read input signals, allow the user to clear display, and set configurations using a switch. Using the ARM MDK this will either be done in the main loop through constant checks or through interrupts. In this way, the system can react to real time inputs.

Requirement: System must make use of at least one digital input.

Description: The system will contain a switch with multiple functionalities. A press clears the LCD display, thus resetting it for a new input sequence, and a toggle up enables/disables automatic scrolling. Using the MDK this will be achieved using the GPIO drivers which facilitate configuring and using inputs such as switches.

Requirement: System must make use of at least one digital output.

Description: The system will make use of two

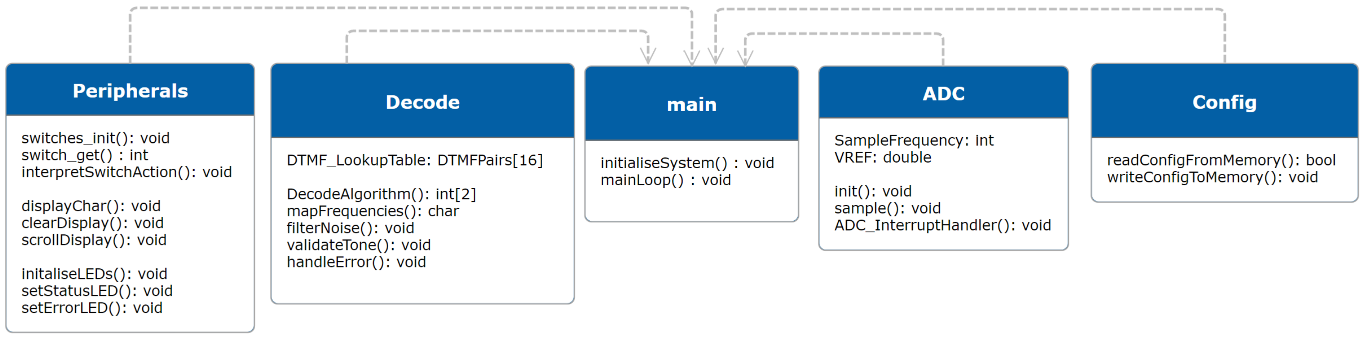

Figure 3: UML Diagram of Required Components / Classes

LEDs (1 red and 1 green) to provide immediate visual feedback of the system status. Under normal and safe operation, the green LED emits light. Upon detection of any error (such as decoding failure) the red LED will instead emit light. This will be done through the MDK's GPIO drivers which provide control to the LEDs' on and off state.

Requirement: System must connect to at least one external peripheral.

Description: A Hitachi HD44780-based 16x2 LCD display will be used to output the decoded result. This can be achieved through manipulating GPIO pins to communicate with the LCD. These are pin P1_24 (used for serial data input), pin P1_20 (acts as the clock signal for synchronizing data transmission) and pin P1_2 (signals the transfer of data from the shift register to the output latches).

Requirement: On power up or reset, system should enter a well-defined safe state.

Description: The aim is that using the MDK, routines will be defined to only respond to valid inputs, thus the system will be designed to never enter an undefined state.

Requirement: System must persistently store configuration settings, independent of the position of physical switches.

Description: The ARM MDK supports interfacing with non-volatile memory devices such as flash memory and EEPROMs. Thus either one will be used in order to keep track of the configurations set by the user.

Requirement: System shall have a useful and appropriate mechanism for indicating error conditions to the user.

Description: As mentioned prior, a red LED will be controlled using the MDK to indicate that the system has encountered an error.

Requirement: System should be robust (incorrect input is handled)

Description: In the event of incorrect input, the system will ignore the input without performing any action, ensuring that it remains in a stable and expected state.

2.5 Steps required for decoding a signal

1. Analogue-to-digital conversion: Continuous analogue signals are sampled using the ADC and stored in the RAM buffer.
2. Noise filtering using convolution: In order to enhance signal a filter is applied such that only frequencies within the DTMF range are allowed.
3. Breaking up the frequency: In order to identify the two frequencies making up the signal, either the Goertzel algorithm or the Fast Fourier Transform will be used.
4. Map frequencies to digits: Once a pair of frequencies has been obtained, they are matched to the corresponding result using a lookup table. Example, the frequency pair 697, 1209 corresponds to the digit '1'.
5. Output Display: The decoded result is then sent directly to the display module's memory such that it is displayed to the user in real-time.

3 Management

3.1 Time Plan and Dependencies

We have identified 11 tasks which need to be tackled. These tasks can be seen in the legend on the left hand side of the Gantt Chart[6] figure (Fig 4).

The required responsiblities were assigned among the members of the group. These responsibilities

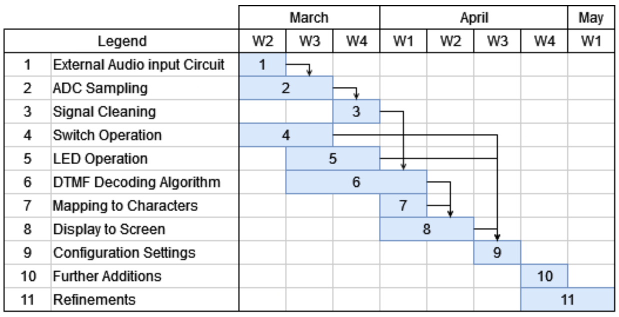

Figure 4: Gantt Chart showing our proposed time plan

may be subject to changes as we proceed further. Below is a breakdown of our planned assignments:

1. Graham Pellegrini is assigned: Primarily the External Audio Input Circuit, Secondarily Signal Cleaning, Thirdly The Decoding Algorithm
2. Luca Vella is assigned: Primarily ADC Sampling, Secondarily Mapping to Characters, LED Operation.
3. Matthew Mifsud is assigned: Primarily Displaying to Screen, Secondarily ADC Sampling, Thirdly Switch Operation.
4. Julian Falzon is assigned: Primarily the Decoding Algorithm, Secondarily Configuration Settings.

By 'Further Additions' we are referring to the implementation of any extra features that are not crucial to the functionality of the core DTMF decoder. On the other hand, by 'Refinements' we mean optimisation and refactoring of the code. Thus, there will be no particular individual assigned for these tasks.

It should be noted that the tasks in the chart were chosen to have a one or two week deadline, with the decoding algorithm taking three weeks, depending on the complexity of the task.

The tasks in the chart are staggered such that they partially depend on the tasks above. The arrows within the chart illustrate which parts require full functionality of a particular task before being able to move forward. For example, a portion of task 2 depends on task 1, this could be the testing of whether input can be read.

3.2 Development Model

The Rapid Software Development model is selected for this project. This model involves dividing the project into smaller, manageable parts that can be developed and tested in brief, iterative cycles. Thus emphasizing speedy development, which is a fitting choice since some decisions may change and evolve. [7]

4 Closure

In conclusion, this design brief has outlined the challenges to be tackled and initiated a plan for solutions to meet all requested requirements. Preliminary specifications for implementation have been set in order to do so. It is important to note that these specifications, such as error handling, the structure of component classes, and other proposed solutions may be restructured or better adapted later on in the implementation timeline. Tasks have also been set out individually, however, a collaborative effort is expected from all members in order to enable more effective problem-solving.

References

[1] Wikipedia contributors, "Dual-tone multi-frequency signaling." https://en.wikipedia.org/w/index.php?title=Dual-tone_multi-frequency_signaling. [Online; accessed 10-March-2024].

[2] W. Furlong, "Safely connect audio line level input to adc." Electrical Engineering Stack Exchange, https://electronics.stackexchange.com/q/625058 (version: 2022-06-26). [Online; accessed 10-March-2024].

[3] Wikipedia contributors, "Voltage divider." https://en.wikipedia.org/w/index.php?title=Voltage_divider. [Online; accessed 10-March-2024].

[4] Vishay Intertechnology, Inc., "1N4148 Datasheet." https://www.vishay.com/docs/81857/1n4148.pdf. [Online; accessed 12-March-2024].

[5] SparkFun Electronics, "HD44780 Datasheet." [Online; accessed 12-March-2024].

[6] Wikipedia contributors, "Gantt chart." "https://en.wikipedia.org/wiki/Gantt_chart". [Online; accessed 9-March-2024].

[7] Kissflow, "Rapid application development." "https://kissflow.com/application-development/rad/rapid-application-development/". [Online; accessed 13-March-2024].